# Implementasi Perangkat Lunak Penyandian Pesan Menggunakan Algoritma RSA

Septya Maharani
Fahrul Agus

*Program Studi Ilmu Komputer, FMIPA Universitas Mulawarman*
*Jl. Barong Tongkok Kampus Gunung Kelua Sempaja Samarinda 75119*

**Abstrak**
Perkembangan teknologi telekomunikasi dan penyimpanan data dengan menggunakan komputer memungkinkan pengiriman data jarak jauh menjadi relatife cepat dan murah. Pada sistem jaringan komputer yang luas seperti internet, maka pengiriman pesan lewat email memungkinkan pesan dapat dibajak oleh orang yang tidak berwenang.
Kemajuan dan perkembangan kriptografi turut mempengaruhi perkembangan aplikasi pengirimian pesan yang disandikan. Kriptografi menjaga kerahasiaan informasi yang terkandung dalam pesan sehingga informasi tersebut tidak dapat diketahui oleh pihak yan tidak sah. Salah satu Algoritma Asimetri yang merupakan jenis algoritma kriptografi yaitu algortima RSA.
Penelitian ini bertujuan untuk mengimplementasikan rancangan sistem penyandian pesan untuk menjadi perangkat lunak bantu yang sesungguhnya. Dokumen rancangan sistem ini telah dipublikasikan pada tulisan yang dimuat pada jurnal yang sama edisi sebelumnya.
Tulisan kali ini memuat hasil implementasi sistem dalam bentuk algoritma pembangkitan kunci, algoritma engkripsi dan dekripsi, integrasi dengan aplikasi pengiriman mail, serta tampilan antar muka perangkat lunak tersebut.

**Kata kunci:** Implementasi, Pembangkitan Kunci, Engkripsi, Dekripsi.

## Latar Belakang

Perkembangan teknologi telekomunikasi dan penyimpanan data dengan menggunakan komputer memungkinkan pengiriman data jarak jauh menjadi relatif cepat dan murah. Di lain pihak pengiriman data jarak jauh melalui gelombang radio, maupun media lain yang digunakan masyarakat luas (*public*) sangat memungkinkan pihak lain dapat menyadap dan mengubah data yang dikirimkan.

Pada sistem jaringan komputer yang luas seperti internet, khususnya pengiriman pesan lewat *email* memungkinkan pesan dapat dibajak oleh orang yang tidak berwenang. Pada masalah keamanan pesan, terdapat dua permasalahan utama yang mesti diperhatikan oleh pengguna yaitu masalah privasi *(privacy)* dan keautentikan *(authentication)*. Privasi mengandung arti bahwa pesan yang dikirimkan hanya dapat dimengerti informasinya oleh penerima yang sah, sedangkan keautentikan mencegah pihak ketiga untuk mengirimkan pesan yang salah atau mengubah pesan yang dikirimkan. Untuk mengatasi masalah keamanan pesan tersebut diperlukan keamanan pesan yang bisa menutupi kelemahan pesan tersebut.

## Tujuan

Penelitian ini bertujuan untuk membuat aplikasi perangkat lunak penyandian pesan menggunakan algoritma RSA. Namun lingkup tulisan ini meliputi implementasi rancangan sistem penyandian pesan untuk menjadi perangkat lunak bantu yang sesungguhnya.

## Algoritma Pembangkitan Kunci

Menemukan dua bilangan prima besar misal *p* dan *q* yang didapat dengan mencoba serangkaian bilangan acak. Pada peroses pembangkitan kunci, terdapat dua pilihan yaitu menggunakan kunci acak dan membuat kunci yang dikehendaki.

Selain itu pula, jika *p*-1 atau *q*-1 memiliki faktorisasi bilangan prima yang kecil, *n* dapat difaktorkan secara mudah dan nilai-nilai dari *p* atau *q* dapat diacuhkan.

Pengguna seharusnya tidak melakukan metode pencarian bilangan prima yang hanya akan memberikan informasi penting tentang bilangan prima tersebut kepada penyerang karena pembangkit bilangan acak yang baik akan memulai nilai bilangan yang digunakan. Pencarian bilangan prima tersebut dilakukan secara acak dengan hasil tidak terduga. Berikut ini mungkin tidak memenuhi kriteria, sebuah bilangan mungkin dapat dipilah dari proses acak tetapi jika bilangan itu mudah untuk ditebak atau

diduga (atau mirip dengan bilangan yang mudah ditebak), maka metode tersebut akan kehilangan kemampuan keamanannya. Apabila bilangan prima telah teracak namun masih dipublikasikan, hal ini akan mempermudah penyerang dalam mendapatkan bilangan tersebut. Jika penyerang dapat menebak separuh dari digit *p* atau *q*, para penyerang dapat dengan cepat menghitung separuh yang lainnya.
Proses pembangkitan kunci dapat dijelaskan prosesnya sebagai berikut:

***Langkah 1 :*** Pembangkitan kunci *p* dan *q* menggunakan proses bilangan acak, sehingga angka akan terbentuk sendiri dengan rumus *genPrima = Rnd * 100* karena bilangan yang digunakan adalah bilangan bulat sehingga agar bilangan bukan bentuk desimal maka dikalikan dengan 100.

***Langkah 2 :*** Pembentukan kunci *n* didapat dari rumus $n = p \times q$, dimana *n* berfungsi untuk menghitung pada proses modulo enkripsi dan dekripsi.

***Langkah 3 :*** Untuk mencari *e*, diperlukan semua bilangan prima yang lebih besar daripada kunci *p* dan *q* dan relatif prima terhadap *m*.

***Langkah 4 :*** Sedangkan pada kunci *d*, digunakan rumus $m = (p-1) \times (q-1)$, dengan menggunakan *m*, akan ditemukan kunci *d* , dengan kekongruenan $ed \equiv 1 \pmod m$, atau $d=1+(k \times m)/ e$.

```
Algoritma Pembuatan Kunci
Deklarasi
P,q, teta, logn : LongInteger

Deskripsi
Read p ← genPrima()
  q ← genPrima()
  n ← p * q
  teta ← (p-1)*(q-1)
  logn  ← logn (n)\log (2)
ulang:
  Do
       e ← cari_e(p, q, n)
       Loop Until FPB(e, teta)
       d ← Multinverse (e,teta)
       if (d < logn or e = d) Then GoTo ulang
endif
write (p, q, n, e, d)
```

Setelah semua proses pembangkitan kunci telah selesai, maka dengan mudah melakukan proses enkripsi dan dekripsi yang secara acak.

**Algoritma Proses Enkripsi**

Proses enkripsi merupakan komponen yang memegang peranan penting dalam arsitektur perangkat lunak ini. Dengan kata lain, mesin utama dalam penyandian ini terdapat pada proses engkripsi ini. Adapun algoritma proses enkripsi sebagai berikut:

***Langkah 1 :*** Membangkitkan kunci publik untuk mengenkripsi pesan. Terlebih dahulu mencari dua buah bilangan prima besar yang acak dan berbeda *p* dan *q*, masing-masing berukuran sama. Kemudian menghitung $n = pq$ dan $m = (p-1)(q-1)$ dan memilih bilangan bulat acak *e* dengan $1 < e < m$, sehingga $gcd(e, m) = 1$. Menggunakan algoritma *extended euclidean* untuk menghitung bilangan bulat unik *d* di mana $1 < d < m$, sehingga $e\ d \equiv 1 \pmod m$. Dengan demikian kunci publik adalah $(n, e)$ dan kunci pribadi adalah *d*.

***Langkah 2 :*** Merubah *plaintext* sesuai dengan desimal ascii-nya. Setelah berbentuk desimal ascii, potong per digit *(n-1)* sehingga berkelompok menjadi beberapa kelompok blok.

***Langkah 3 :*** Secara beruntun konversikan masing-masing blok-blok ini adalah $pl_1$, $pl_2$,....$pl_n$ ke rumus $c = pl^e \bmod n$, dimana $e$ merupakan eksponen yang relatif prima dengan $m$ dan $n$ merupakan modulus.

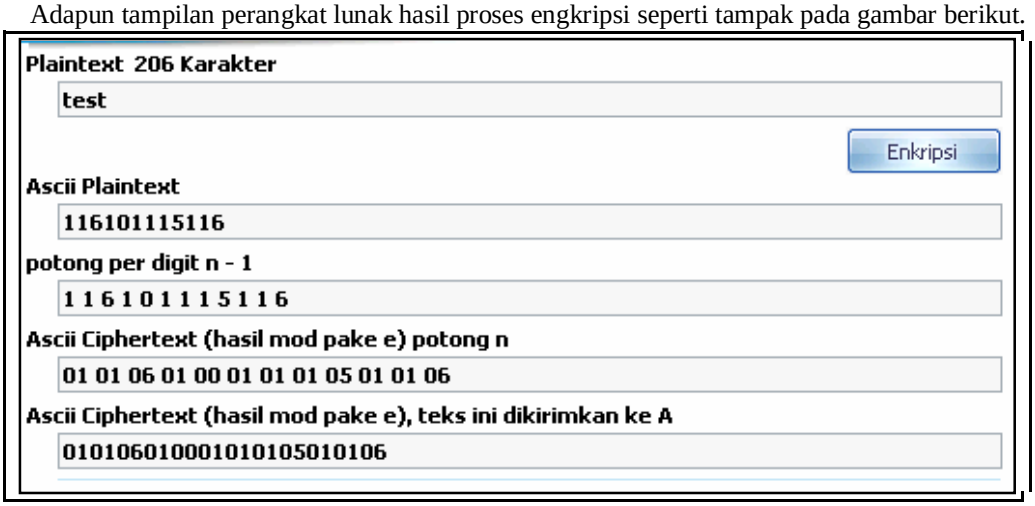
***Langkah 4 :*** Setiap blok akan berubah menjadi $c_1$, $c_2$,...$c_{n,,}$ hasil akhir enkripsi adalah teks *plaintext* berubah menjadi *chipertext.*

Jika langkah-langkah tersebut diwujudkan dalam bentuk algoritma psedocode maka menjadi seperti gambar berikut.

```
Algoritma Enkripsi

Deklarasi
  pn,m,pe,jm, jlhhrf, i, jlhascii       : byte
  strAsli, hrf, aschrf, ascii_blok      : String
  blok_tampil, blok_enkrips, cipher_blok: String
  cipher_gab, aschrf_gab, blok_enkrip   : String

Deskripsi
read (strAsli)
    If strAsli <> "" Then
       pn ← Len(CStr(n))
       jlhhrf ← Len(strAsli)

       For i = 1 To jlhhrf
            hrf ← Mid(strAsli, i, 1)
            aschrf ← Asc(hrf)
         If Len(aschrf) = 2 Then aschrf = 0 and aschrf
            aschrf_gab ← aschrf_gab and aschrf
            ascii_blok ← ascii_blok and aschrf and " "
        Next i
        m ← pn - 1
        jlhascii ← Len(aschrf_gab)

        jm ← jlhascii Mod m
        If jm <> 0 Then
            For i = 1 To jm
                aschrf_gab ← aschrf_gab and "0"
                jlhascii ← jlhascii + 1
            Next i
        endif


        For i = 1 To jlhascii Step m
            blok ← Mid(aschrf_gab, i, m)
            pb ← Len(CStr(blok))
            bloks ← CStr(blok)
            Do While (pb Mod m <> 0)
                pb ← pb + 1
                bloks←"0" and bloks
        Loop
           blok_tampil ← blok_tampil and bloks and " "
```

```
            blok_enkrip ⟵ modExp(blok, e, n)
           blok_enkrips ⟵ CStr(blok_enkrip)
           pe ⟵ Len(blok_enkrips)
        Do While (pe Mod pn <> 0)
           pe ⟵ pe + 1
           blok_enkrips ⟵ "0" and blok_enkrips
           cipher_blok ⟵cipher_blok and blok_enkrips
    and " "
           cipher_gab ⟵ cipher_gab and blok_enkrips
        Next i
        endif
```

Adapun tampilan perangkat lunak hasil proses engkripsi seperti tampak pada gambar berikut.




Gambar 1. Tampilan Proses Enkripsi

**Algoritma Proses Dekripsi**

Proses dekripsi adalah komponen yang berfungsi mengembalikan *plaintext* menjadi *chipertext* yang dapat dipahami penerima pesan. Setelah pengguna mendapatkan pesan *email* dari pengirim pesan, maka pesan dapat di *paste* dan dimasukkan ke kolom chipertext untuk proses dekripsi. Pada proses dekripsi, penting bagi pengguna untuk menggunakan kunci yang digunakan pengirim. Proses dekrispi ini dinamakan juga penguraian kembali teks menjadi teks asal sebelum disandikan. Komponen dekripsi menggunakan algoritma sebagai berikut :

***Langkah 1 :*** *Chpertext* yang telah dikelompokan beberapa blok dapat di uraikan menjadi $c_1, c_2,...,c_n$.

***Langkah 2 :*** penghitungan *chipertext* menggunakan kunci privat dengan rumus $pl = c^d \bmod n$, dimana *d* merupakan kongkruen lanjar yang berupa bilangan bulat dengan rumus $ed \equiv 1 \pmod m$.

***Langkah 3 :*** Seteleh setiap blok *chipertext* dihitung dengan rumus pada langkah 2, telah didapatkan blok desimal ascii *plaintext* kembali.

***Langkah 4 :*** Blok desimal Ascii akan kembali bersatu tanpa adanya pengelompo-kan blok per digit *(n-1).*

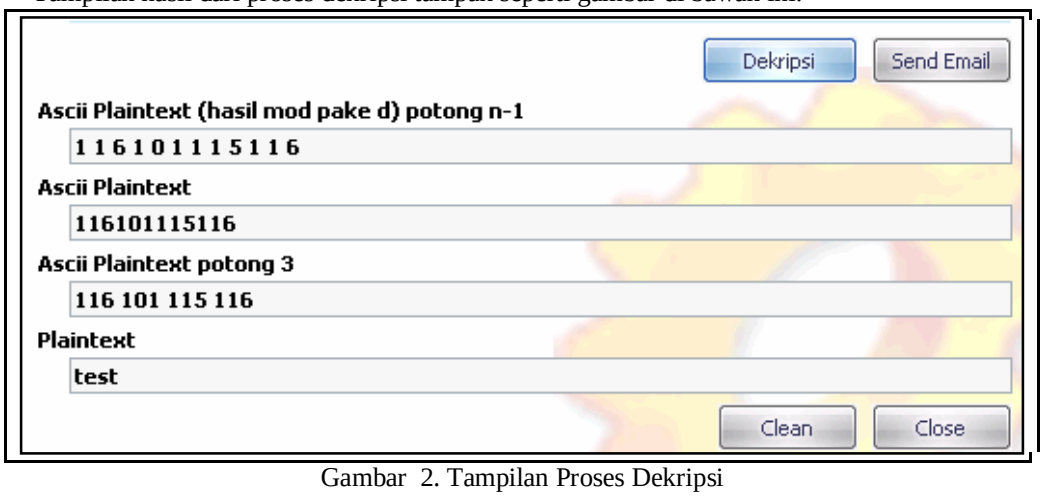
***Langkah 5 :*** Berubahnya desimal ascii menjadi teks biasa *(plaintext)* yang telah mengalami proses dekripsi.

Algoritma proses dekripsi dalam bentuk psedocode seperti dituliskan pada teks berikut.

```
Algoritma Dekripsi
Deklarasi
Pn,m,pd                          : byte
Jlhascii, blok_dekrip, pbdg      : integer
blok_dekrips, blok_dekrip_gab    : String
aschrf3, aschuruf, plain         : String
```

```
    Deskrispi
        pn ←— Len(CStr(n))
        m  ←— pn - 1
        jlhascii ←— Len(txtAscicipher)
        For i = 1 To jlhascii Step pn
            blok_dekrip ←— Mid(txtAscicipher, i, pn)
            blok_dekrip ←— modExp(blok_dekrip, d, n)
            blok_dekrips ←— CStr(blok_dekrip)
            pd ←— Len(blok_dekrips)
         Do While (pd Mod m <> 0)
                pd ←— pd + 1
                blok_dekrips ←— "0" and blok_dekrips
         Loop

           blok_blok ←— blok_blok and blok_dekrips and " "
           blok_dekrip_gab ←— blok_dekrip_gab and
           blok_dekrips
        Next i
            pbdg ←— Len(blok_dekrip_gab)
        Do While pbdg Mod 3 <> 0
            pbdg ←— pbdg - 1
            blok_dekrip_gab ←— Left(blok_dekrip_gab, pbdg)
        Loop

        For I = 1 To pbdg Step 3
            aschuruf ←— Mid(blok_dekrip_gab, i, 3)
            aschrf3 ←— aschrf3 and aschuruf and " "
            huruf ←— Chr(aschuruf)
            plain ←— plain + huruf
        Next i
        Endif
```

Tampilan hasil dari proses dekripsi tampak seperti gambar di bawah ini:

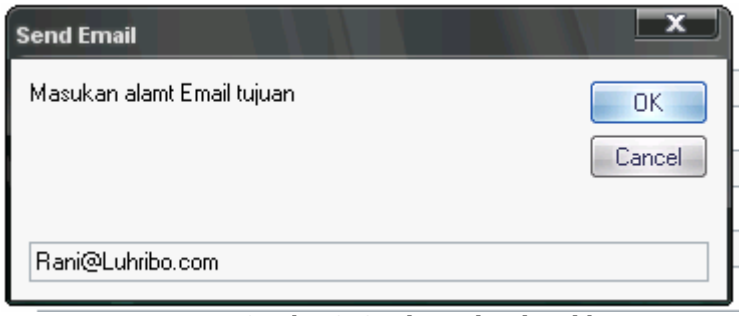



Gambar 2. Tampilan Proses Dekripsi

**Integrasi dengan Pengiriman Mail**

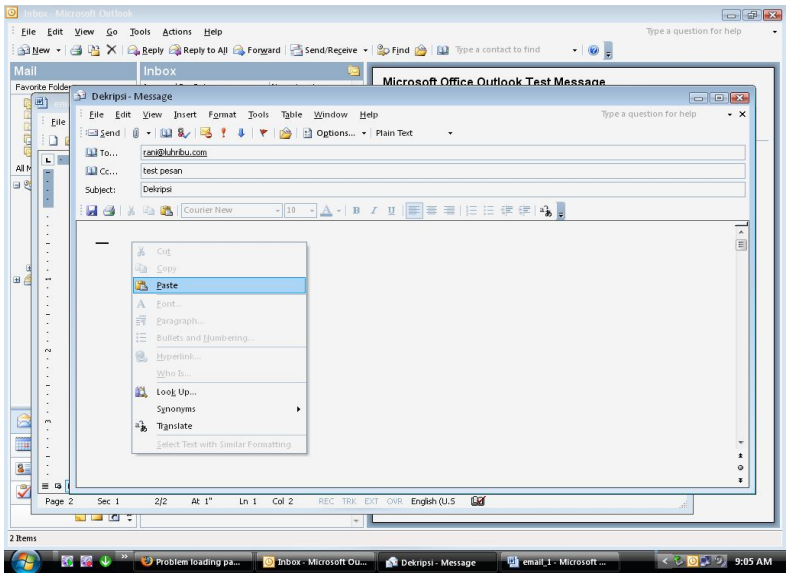
Sebagai perangkat lunak yang berfungsi untuk penyandian pesan, maka penyandian ini bisa digunakan untuk mengirim pesan penting lewat *e-mail*. Pada proses *Send E-Mail* ini, ini aplikasi diintegrasikan dengan *tools Microsoft Outlook*.

Hasil dari proses enkripsi, dimana data atau pesan masih bersifat *chipertext* akan di *Copy* dan dikirim ke *e-mail*, sehingga pesan dapat sampai ketujuan dalam keadaan pesan yang tersandikan.

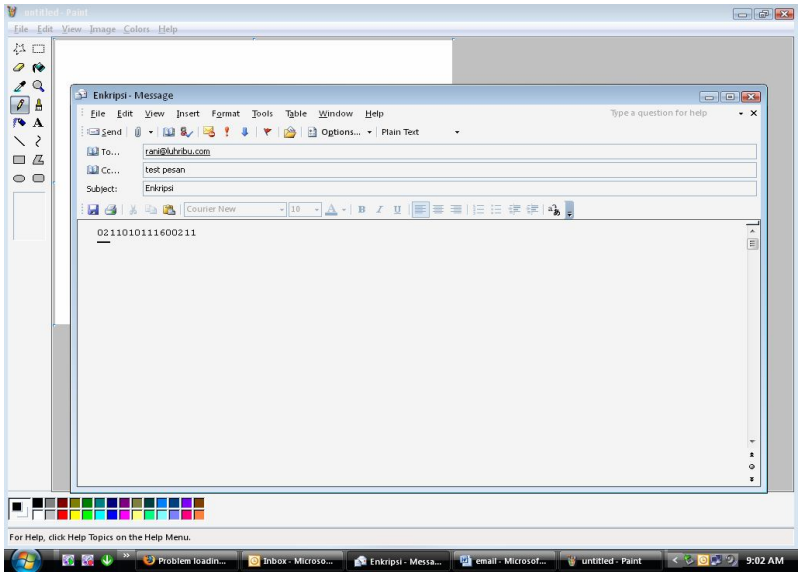



Gambar 3. Send Email pada aplikasi

Setelah menulis alamat *email* yang akan dituju, maka akan tersambung ke *Microsoft Outlook 2003* yang telah menyediakan layanan pengiriman email. Pesan yang akan dikirim telah berbentuk *chipertext*, seperti pada gambar dibawah ini:

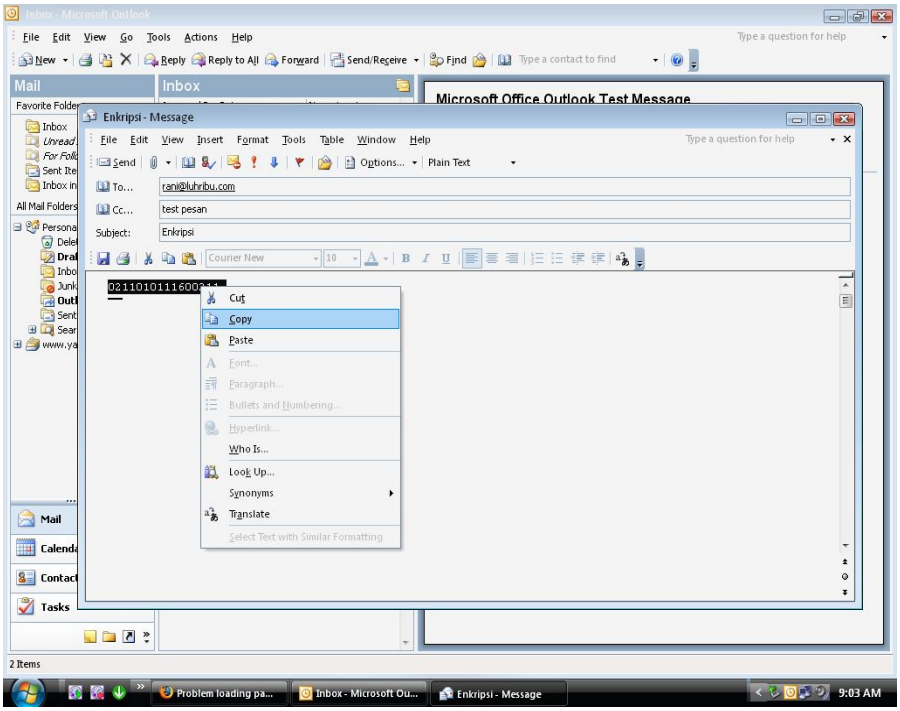



Gambar 4. Paste *Chipertext*




Gambar 5. Mengirim *Chipertext*

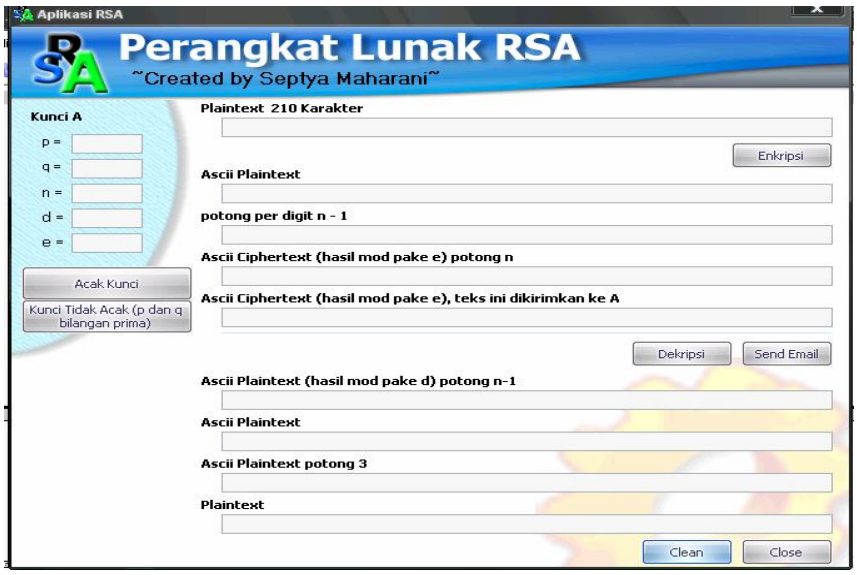
Setelah pesan dimasukan kedalam *email*, pesan akan dikirim dan siap untuk didekripsi oleh penerima pesan dengan cara meng-*copy* pesan dan *paste* kedalam perangkat lunak dengan memasukan kunci yang digunakan untuk mengenkripsi pesan.




Gambar 6. Pesan yang telah dienkripsi

**Tampilan Perangkat Lunak**

Seluruh algoritma yang ada diwujudkan menjadi sebuah aplikasi dengan menggunakan perangkat lunak pengembangan Visual Basic. Hal itu diwujudkan dalam sebuh project yang terdiri atas beberapa form. Adapun tampilan form utama dari aplikasi penyandian ini seperti pada gambar berikut.




Gambar 7. Tampilan Perangkat Lunak

**Kesimpulan**

Pada penelitian ini telah dihasilkan sebuah perangkat lunak bantu penyandian pesan menggunakan algoritma RSA. Perangkat lunak ini juga terintegrasi dengan aplikasi pengiriman pesan, sehingga dapat memudahkan para pengguna untuk menyandikan pesannya ketika akan dikirimkan melalui elektronik mail.

**Daftar Pustaka**

1. Ahmad Basuki. *Modeling dan Simulasi*, IPTAQ Mulia Abadi, Jakarta, 2004.
2. Abraham Sinkov, 1996. *Elementary Crytanalisis a mathematical Approach. Matematical, Association of Amerika.*
3. *Renaldi munir, 1999. Algoritma dan Pemrograman. Informatika Bandung, Bandung.*
4. *Wiryanto Dewobroto. Aplikasi Sain dan Tejnik dengan Visual Basic 6.0, Lippi Karawaci, 2003.*
5. Yusuf Kurniawan, Ir. MT, 2004. *Kriptografi keamanan internet dan jaringan komunikasi.* Informatika Bandung, Bandung.